第4回 統計・機械学習若手シンポジウム

深層生成モデルと世界モデル

東京大学工学系研究科　技術経営戦略学専攻

松尾研究室　特任研究員

鈴木雅大

2019/11/15

自己紹介

鈴木雅大（東京大学松尾研究室　特任研究員）

- 経歴
  - 2015年3月 北海道大学情報科学研究科修了
  - 2018年3月 東京大学工学系研究科修了
    - 博論：深層学習と生成モデルによるマルチモーダル学習に関する研究
  - 2018年4月~ 東京大学工学系研究科 特任研究員
- 研究分野：
  - 深層生成モデル
  - マルチモーダル学習
  - 転移学習（ゼロショット学習）
- 活動など：
  - Goodfellow, Bengioら著「深層学習」の監訳・分担翻訳

目次

- 深層生成モデル
- 深層生成モデルとマルチモーダル学習
- 世界モデルと深層生成モデル
- まとめ

深層生成モデル

深層生成モデル

- 深層学習の研究分野では，深層生成モデルの研究が進んでいる．
  - 生成系（画像や文書）の他に，異常検知，半教師あり学習，表現学習，メタ学習など

Figure 1: Class-conditional samples generated by our model.

[Brock+ 18]

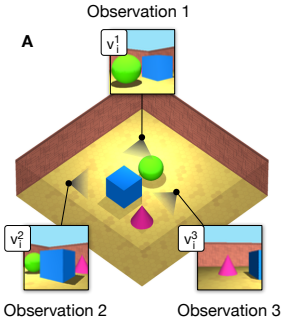
A stop sign is flying in blue skies.

[Mansimov+ 15]

i went to the store to buy some groceries .
i store to buy some groceries .
i were to buy any groceries .
horses are to buy any groceries .
horses are to buy any animal .
horses the favorite any animal .
horses the favorite favorite animal .
horses are my favorite animal .

[Bowman+ 15]

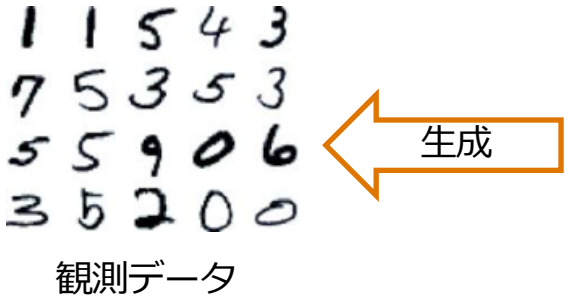
[Zhu + 17]



[Eslami+ 18]

生成モデル

- （確率的）生成モデル：
  - データとして観測される観測変数が何らかの確率モデルから生成されていると仮定し，その生成過程を確率分布によってモデル化するアプローチ．
    - 観測変数の背景にある因子（確率変数）として潜在変数も仮定することが多い．
  - 「データがどのようにできているか？」を明示的に表すことができ，モデルからデータを生成することができる．



観測データ

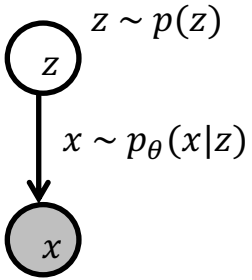
生成

観測変数

潜在変数

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz$$

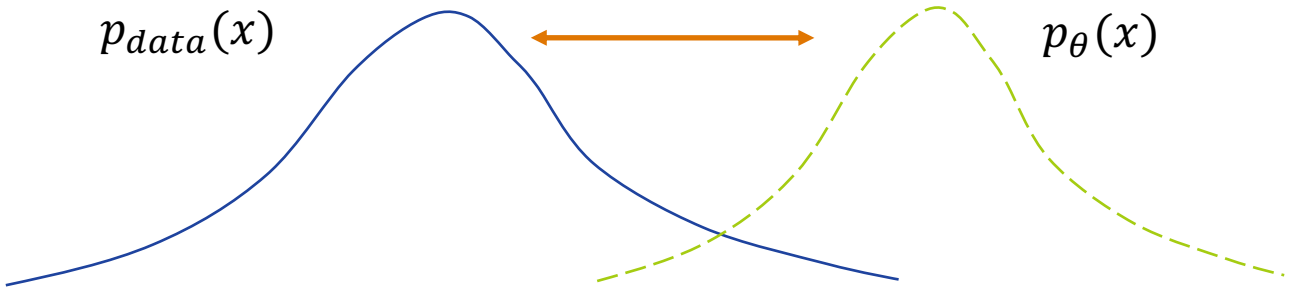
パラメータ



生成モデル

生成モデルの学習

- 真の分布（データ分布）$p_{data}(x)$を近似するモデル分布（生成モデル）$p_\theta(x)$を学習したい.
  - 生成モデルが真の分布を近似するときのθを求めたい.



- どのような基準で分布間の距離を測ればいいか？
  - 真の分布は実際にはわからないことにも注意.

生成モデルの学習

- カルバック・ライブラー（Kullback–Leibler，KL）ダイバージェンスで違いを測る．

$$D_{KL}[p_{data}(x)||p_\theta(x)]$$
$$= E_{p_{data}(x)}[\log \frac{p_{data}(x)}{p_\theta(x)}] = E_{p_{data}(x)}[\log p_{data}(x)] - E_{p_{data}(x)}[\log p_\theta(x)]$$

対数尤度関数

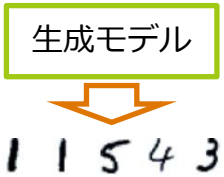
  - これを最小化するパラメータを求める．

- 上式より，対数尤度関数の期待値を最大化するパラメータを求めれば良い．
  - 真の分布は分からないので，サンプル近似したもの（つまり訓練集合$\mathcal{D} = \{x_i\}_{i=1}^N$での平均）を目的関数とする（最尤推定）．

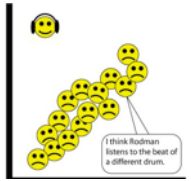
$$E_{p_{data}(x)}[\log p_\theta(x)] \cong \frac{1}{N} \sum_{x_i \in \mathcal{D}} \log p_\theta(x_i)$$

生成モデルでできること

- 生成：
  - 生成モデルが学習できれば，新たなデータを生成できる．
  - 「生成」モデルと呼ばれるのはここから



- 密度推定：
  - データを入力すると，それがどれだけ生成モデルと違うかがわかる．
  - 外れ値検出や異常検知に用いられる．



http://jblomo.github.io/datamining290/slides/2013-04-26-Outliers.html

- 欠損値補完，ノイズ除去：
  - 欠損やノイズのある入力を入れると，補完してくれる．

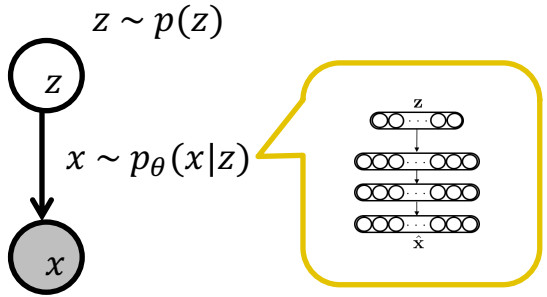
深層生成モデル

- 観測変数が複雑な場合，単純な確率分布では直接表現できない.
  - 特に観測変数がベクトルで，その要素（次元）間の依存関係が非線形な場合（高解像度の画像など）

非線形な関係性を表すには？-> 深層ニューラルネットワーク（DNN）

- 深層生成モデル（deep generative model）
  - 確率分布をDNNで表現した生成モデル.
  - DNNによって，複雑な入力をend-to-endに扱えるようになった.

生成モデルによって明示的に生成過程をモデル化できる
+
DNNによって非線形な関係性を捉えられる



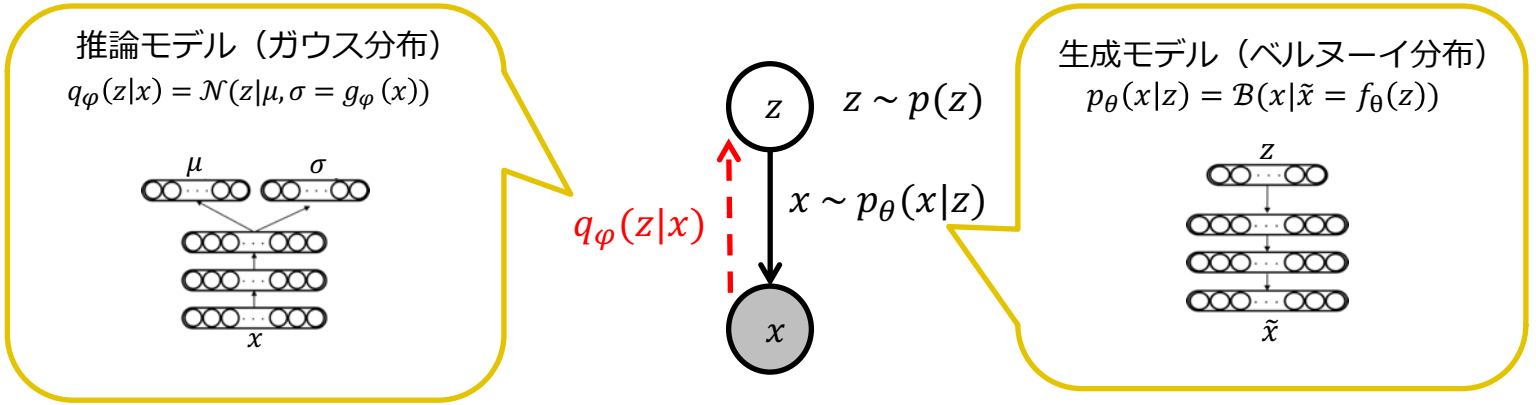
深層ニューラルネットワークによる確率分布の表現

深層生成モデルの種類

	学習するモデル	生成モデルの尤度計算	生成	推論
VAE	生成モデル： $p(x,z) = \int p(x\|z)p(z)dz$ 推論モデル： $q(z\|x)$	直接は不可能（対数尤度の変分下界（ELBO）が計算可能）	低コスト	可能（推論モデル）
GAN	生成器： $G(z)$ 識別器： $D(x)$	不可能（識別器が真のモデルとの尤度比を推定）	低コスト	不可能（手法による）
自己回帰モデル	複数の条件付きモデル： $p(x) = \prod_i^D p(x_i\|x_1, \dots, x_{i-1})$	可能	高コスト	潜在変数がない
フローベースモデル	フロー（可逆な関数）： $x = f(z)$	可能	低コスト	可能（逆変換）
エネルギーベースモデル	エネルギー関数：$E(x)$	困難（分配関数の計算が必要）	高コスト	潜在変数を持つモデルの場合は可能

Variational Autoencoder

- Variational autoencoder（VAE） [Kingma+ 13, Rezende+ 14]
  - 潜在変数zをもつ観測変数xの深層生成モデル
  - 潜在変数zへの推論を，観測xを入力としたDNNで表現（amortized inference）

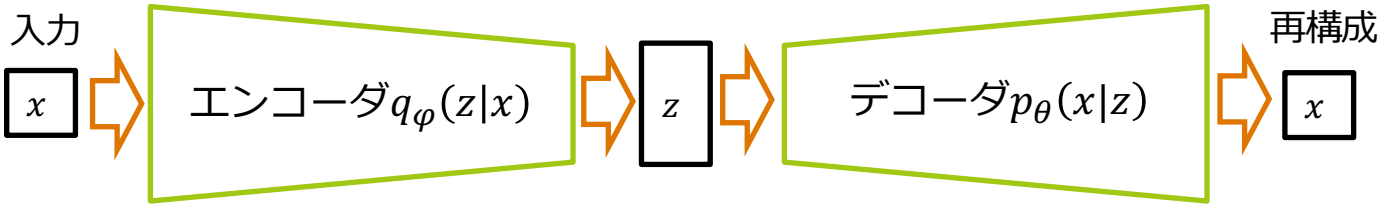


Variational Autoencoder

- 目的関数：対数尤度の変分下界（エビデンス下界，evidence lower bound；ELBO）

$$\log p_\theta(x) \geq \underbrace{E_{q_\varphi(z|x)}[\log p_\theta(x|z)]}_{\text{負の再構成誤差}} - \underbrace{D_{KL}[q_\varphi(z|x) \parallel p(z)]}_{\text{推論の正則化}}$$

- 分布のパラメータ化等の話を省略して，情報の流れを確認．
  - VAEでは推論モデルで入力xをzにエンコードし，生成モデルでzからxをデコード（再構成）する．
    - 推論モデルと生成モデルをオートエンコーダにおけるエンコーダとデコーダとみなせる．



生成画像

- ランダムなzから画像を生成
  - 綺麗に生成できているが，輪郭等がぼやける傾向がある．

[Kingma+ 13]より

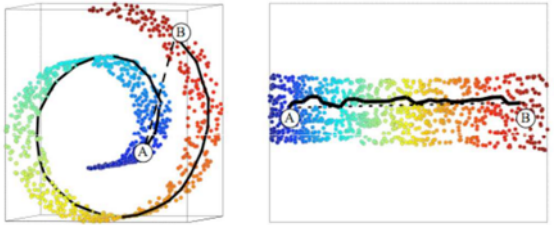
@AlecRad

VAEと表現学習

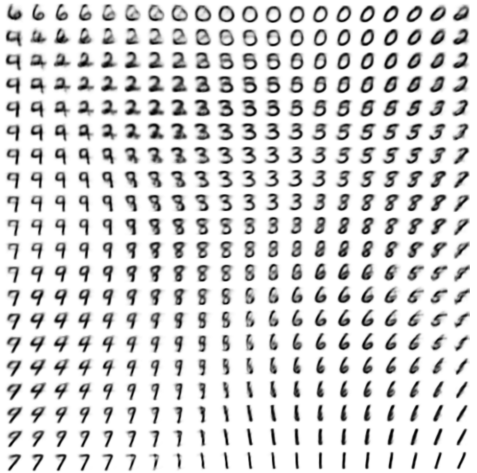
- VAEでは，再構成だけでなく表現zも学習しているとみなせる．
  - エンコーダ$q_\varphi(z|x)$が表現への写像を学習している．
  - VAEは表現学習手法として優れた手法．
- 表現学習：
  - 入力から「良い表現」を（できれば教師なしで）獲得する学習．
  - 「良い表現」とは？
    - 元のデータを保持しつつ，他のタスクにも使い回せるような表現だと良さそう．
- Meta-Prior [Bengio+ 13, Goodfellow+ 16]（言葉自体は[Tschannen+ 18]）
  - 多くのタスクに使える表現の性質に関する仮定
  - 多様体，Disentangle，概念の階層性，半教師あり学習，クラスタ性など

多様体学習

- 多様体：
  - 確率の大部分が集中していて，その集中している部分は局所的に連続している（ユークリッド空間とみなせる）
  - これを低次元空間にうまく展開するように学習する：多様体学習



- VAEによる多様体学習
  - VAEは，表現空間で多様体学習ができる.
  - 2次元の潜在空間の可視化（右図，MNIST）
    - 潜在空間上で移動しそのときの画像を生成



[Kingma+ 13]

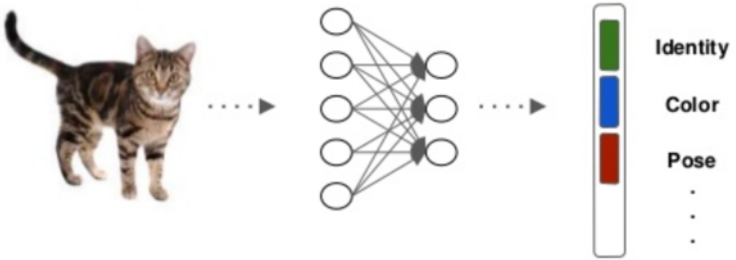
Meta-Priorを実現するVAE

- VAEで表現についての様々なMeta-Priorを実現するために，多くの手法が提案されている（[Tschannen+ 18]より）

Meta-prior	Methods
Disentanglement	β-VAE (6) [2], FactorVAE (8) [3], β-TCVAE (9) [4], InfoVAE (9) [5], DIP-VAE (11) [6], HSIC-VAE (12) [7], HFVAE (13) [8], VIB [9], Information dropout (15) [10], DC-IGN [11], FaderNetworks (18) [12], VFAE (17) [13]
Hierarchical representation[1]	PixelVAE [14], LVAE [15], VLaAE [16], Semi-supervised VAE [17], PixelGAN-AE [18], VLAE [19], VQ-VAE [20]
Semi-supervised learning	Semi-supervised VAE [17], [21], PixelGAN-AE (14) [18], AAE (16) [22]
Clustering	PixelGAN-AE (14) [18], AAE (16) [22], JointVAE [23], SVAE [24]

Disentanglement

- Disentanglement（もつれを解く表現）
  - データは独立に変化する要素から生成されているという仮定
    - 例) 物体の向き，光源の状態



https://www.slideshare.net/lubaelliott/emily-denton-unsupervised-learning-of-disentangled-representations-from-video-creative-ai-meetup

  - 利点：
    - 人間が解釈しやすい表現（「概念」の獲得）
    - 様々なタスクに転用できる可能性
  - 最近では，物理学における対称性（symmetry）と関連づける議論もある [Higgins + 18]

VAEとdisentanglement

- VAEのELBO（再掲）

$$\mathcal{L}(x;\theta,\varphi) = \mathrm{E}_{q_\varphi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\varphi(z|x) \parallel p(z)]$$

  - 第2項の正則化項では$q_\varphi(z|x)$を$p(z)$に近づけるように学習する.

- 通常，$p(z)$には標準ガウス分布$\mathcal{N}(0,\mathbf{I})$が選択される.
  - 各次元は独立（$p(z)=\prod_i p(z_i)$）
  - したがって，この正則化は潜在変数の各要素が独立になるように制約している.

-> disentangleな表現へ

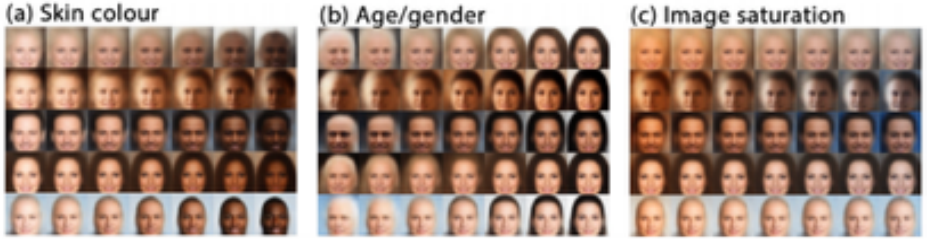
β-VAE

- β-VAE [Higgins+ 17]
  - VAEのELBOの第2項に係数βを導入

$$E_{q_\varphi(z|x)}[\log p_\theta(x|z)] - \beta D_{KL}[q_\varphi(z|x) \parallel p(z)]$$

  βを大きく（$\beta \gg 1$）する.

- すると，潜在変数の各次元で独立な表現（=disentanglement）が獲得される.



※ 実際には，パラメータCを導入した以下の式の方が，よりdisentangleされる [Burgess+ 17]

$$E_{q_\varphi(z|x)}[\log p_\theta(x|z)] - \beta|D_{KL}\left[q_\varphi(z|x) \parallel p(z)\right] - C|$$

正則化項の分解

- ELBOの正則化項（のデータ分布$p_{data}(x)$における期待値）は次のように分解される．

$$E_{p_{data}(x)}\left[D_{KL}\left[q_\varphi(z|x) \parallel p(z)\right]\right] = I(x;z) + D_{KL}\left[q_\varphi(z) \parallel p(z)\right]$$

  - 第1項：xとzの相互情報量$I(x;z)$
  - 第2項：aggregate事後分布$q(z) = \int q(z|x)p_{data}(x)dx$ [Makhzani+ 16]と事前分布$p(z)$のKLダイバージェンス [Hoffman+ 16]
    - Disentanglementに関連する項は第2項

- 「Disentanglement（第2項）」と「zにおけるxの情報量（第1項）」がトレードオフ．
  - Disentangleしようと正則化項の重みを大きくすると，相互情報量が小さくなり，うまく情報がエンコードできなくなる．
  - 表現力の高いデコーダを用いても潜在表現の獲得を無視する傾向がある．

=> posterior collapse

Disentangle VAE

- FactorVAE [Kim+ 18]
  - Aggregate事後分布$q(z)$が独立になるように，全相関による正則化項を追加する．

$$TC\left(q_\varphi(z)\right) = D_{KL}[q_\varphi(z)||\prod_j q_\varphi(z_j)]$$

ただし，この項は評価できないので密度比トリック（後述する敵対的訓練）を用いる．

$$\mathcal{L}_{FactorVAE}(x;\theta,\varphi) = \mathcal{L}_{VAE}(x;\theta,\varphi) - \lambda \cdot TC\left(q_\varphi(z)\right)$$

- PixelGAN-AE [Makhzani+ 17]
  - デコーダにPixelCNNを使うと，潜在表現が無視される（posterior collapse）．
  - 相互情報量$I(x;z)$を足すことで，ELBOの相互情報量の項を打ち消す．

$$\mathcal{L}_{PixelGAN-AE}(x;\theta,\varphi) = \mathcal{L}_{VAE}(x;\theta,\varphi) + I(x;z)$$

そのほかの正則化手法

- [Tschannen+ 18]より

$$\mathcal{L}_{VAE}(x;\theta,\varphi) - \lambda_1\left[R_1(q_\varphi(z|x))\right] - \lambda_2[R_2(q_\varphi(z))]$$

ラベル

WORK	$\mathcal{L}$.	R_1	R_2	Y
β-VAE [2]	VAE	$D_{\mathrm{KL}}(q_\phi(z\|x)\|p(z))$		
VIB [9]	VAE	$D_{\mathrm{KL}}(q_\phi(z\|x)\|p(z))$		O Optional
PixelGAN-AE[18]	VAE	$-I_{q_\phi}(x;z)$		O
InfoVAE [5]	VAE	$D_{\mathrm{KL}}(q_\phi(z\|x)\|p(z))$	$D_{\mathrm{KL}}(q_\phi(z)\|p(z))$	
Info. dropout [10]	VAE	$D_{\mathrm{KL}}(q_\phi(z\|x)\|p(z))$	$\mathrm{TC}(q_\phi(z))$	O
HFVAE [8]	VAE	$-I_{q_\phi}(x;z)$	$R_{\mathcal{G}}(q_\phi(z))+\lambda_2'\sum_{G\in\mathcal{G}} R_{\mathcal{G}}(q_\phi(z))$	
FactorVAE [3, 4]	VAE		$\mathrm{TC}(q_\phi(z))$	
DIP-VAE [6]	VAE		$\|\mathrm{Cov}_{q_\phi(z)}[z] - I\|_{\mathrm{F}}^2$	
HSIC-VAE [7]	VAE		$\mathrm{HSIC}(q_\phi(z_{G_1}), q_\phi(z_{G_2}))$	O
VFAE [13]	VAE		$\mathrm{MMD}(q_\phi(z\|s=0), q_\phi(z\|s=1))$	✓ 必要
DC-IGN [11]	VAE			✓
FaderNet. [12]; [37][2]	AE	$-\mathbb{E}_{\hat{p}(x,y)}[\log P_\psi(1-y\|E_\phi(x))]$		✓
AAE/WAE [22, 36]	AE		$D_{\mathrm{JS}}(E_\phi(z)\|p(z))$	O

- 高い尤度のモデルは必ずしも「良い」表現の獲得を意味しないことも指摘されている[Alemi+ 18]

なぜVAEの生成画像はぼやけてしまうのか？

- VAEでは，データが与えられた下での尤度を計算する必要がある．

$$\mathrm{E}_{q_\varphi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\varphi(z|x) \parallel p_\theta(z)]$$

生成モデルの尤度

  - 生成モデルの分布の選択によって再構成誤差の種類が制約されることを意味する．
    - e.g., 二乗誤差（ガウス分布），交差エントロピー誤差（ベルヌーイ分布）

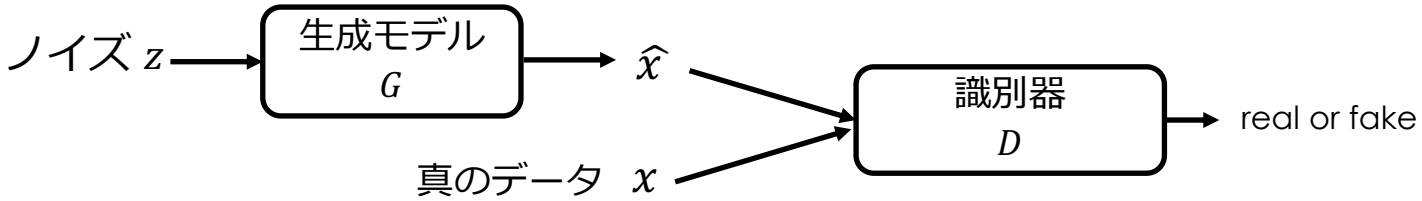
-> 明示的なモデル

- 限定された誤差関数を用いるのではなく，誤差関数自体を学習で求めればいいのでは？

-> 暗黙的なモデル[Mohamed+ 16]

Generative adversarial network

- Generative adversarial network（GAN） [Goodfellow+ 2014]
  - 学習可能な識別器$D(x)$ を導入する.
    - 生成データが真かどうかを判定することで，生成モデルの評価を行う.



- GとDで次のゲームをする（敵対的訓練）.
  - GはなるべくDを騙すようにxを生成する（生成モデルの学習）
  - DはなるべくGに騙されないように識別する（生成モデルの評価の学習）

->最終的には，Dが本物と区別できないようなxがGから生成される.

GANによる画像生成の進展

Reproduced from a Goodfellow's tweet
https://twitter.com/goodfellow_ian/status/1084973596236144640

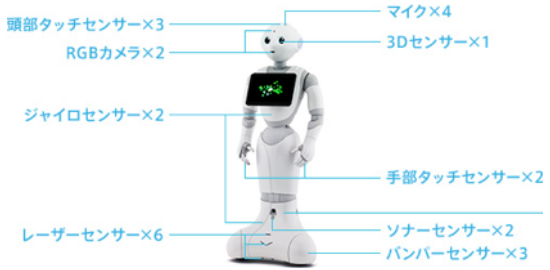
深層生成モデルの種類（再掲）

	学習するモデル	尤度の計算	生成	推論
VAE	生成モデル： $p(x,z) = \int p(x\|z)p(z)dz$ 推論モデル： $q(z\|x)$	直接は不可能（対数尤度の変分下界（ELBO）が計算可能）	低コスト	可能（推論モデル）
GAN	生成器： $G(z)$ 識別器： $D(x)$	不可能（識別器が真のモデルとの尤度比を推定）	低コスト	不可能（手法による）
自己回帰モデル	複数の条件付きモデル： $p(x) = \prod_i p(x_i\|x_1, \dots, x_{i-1})$	可能	高コスト	潜在変数がない
フローベースモデル	フロー（可逆な関数）： $x = f(z)$	可能	低コスト	可能（逆変換）
エネルギーベースモデル	エネルギー関数：$E(x)$	困難（分配関数の計算が必要）	高コスト	潜在変数を持つモデルの場合は可能

深層生成モデルとマルチモーダル学習

マルチモーダル学習

- 我々はマルチモーダル情報を取り入れることで，単一のモダリティ情報よりも確実な情報処理を行っている．
- ロボットも複数のセンサから様々な種類の情報を獲得している
  - 動画，音声，角度や加速度情報，距離情報など



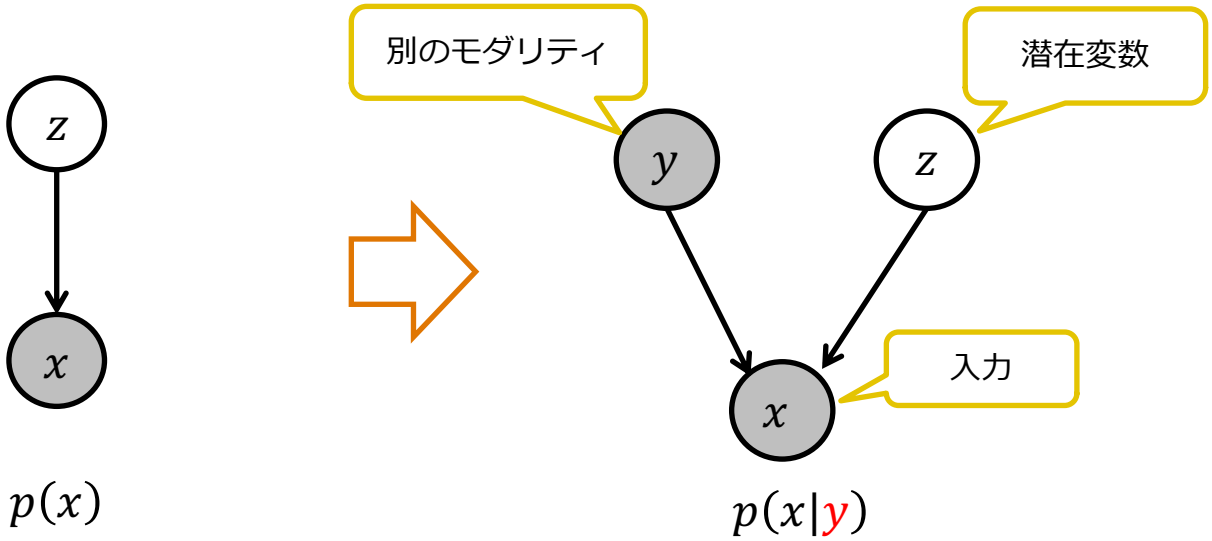
https://www.softbank.jp/robot/consumer/products/spec/

- 機械学習においても，マルチモーダルデータを活用して判断・予測を行いたい．

⇒ マルチモーダル学習

条件付け深層生成モデル

- 観測変数y （xと異なるモダリティ）で条件づけた深層生成モデル
  - y から xへの生成過程を表現.



例：文書から画像の生成

- 文書yから画像x の生成
  - [Mansimov+ 15]


A stop sign is flying in blue skies.


A herd of elephants flying in the blue skies.

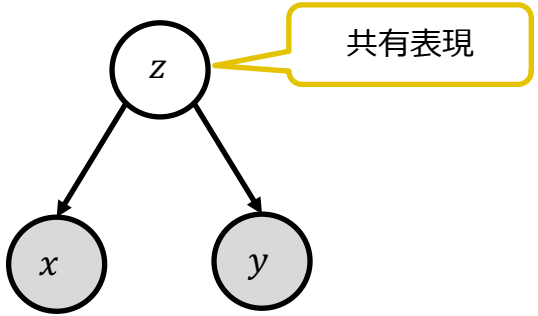

A toilet seat sits open in the grass field.


A person skiing on sand clad vast desert.

  - [Zhang+ 16]

深層生成モデルによる同時モデル

- 異なるモダリティの同時分布$p(x, y)$をモデル化
  - 適切に学習できれば，任意の条件付けを行って生成できるはず（双方向生成，$p(x|y), p(y|x)$）
  - 潜在変数は2つのモダリティを統合した表現（共有表現）を獲得できるはず．

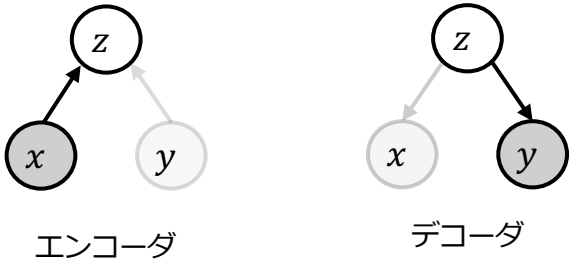


$$p(x, y) = \int p(x|z)p(y|z)p(z)dz$$

- VAEで容易に実現可能
  - モダリティごとにデコーダを増やす．

欠損モダリティ問題

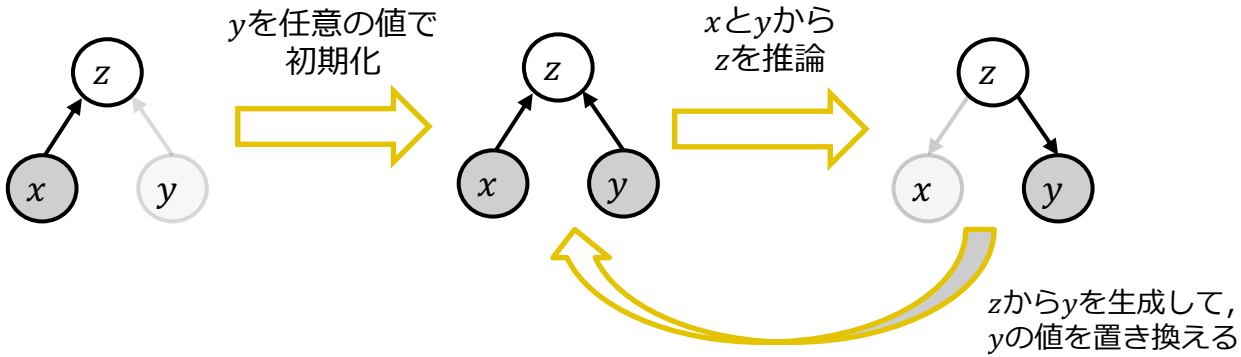
- モダリティ間を双方向変換するときは，片方のモダリティを欠損させる．
  例：xからyへの変換



- 欠損させるモダリティyの情報量が大きい場合，xだけでは適切にzを推論できずに崩れてしまう可能性がある（欠損モダリティ問題）．
  - zが適切に推論できなければ，yも適切に生成できなくなる．

既存手法：反復サンプリング手法

- VAEにおける欠損値補完
  - 反復的に欠損した部分（モダリティ）を補完する（反復サンプリング手法[Rezende 14]）



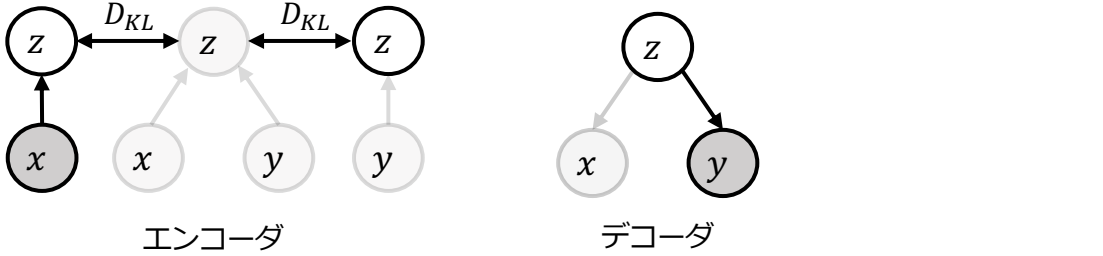
しかし欠損値の情報量が大きい場合，この手法では補完しきれないことを確認.

提案手法：JMVAE

- 単一モダリティのエンコーダ$q(z|x)$，$q(z|y)$を用意して，元のエンコーダ$q(z|x,y)$を近似する．
  - VAEの目的関数に近似のための項（KLダイバージェンス）を加える．

$$\mathrm{E}_{q_\varphi(z|x,y)}[\log p_\theta(x,y|z)] - D_{KL}[q_\varphi(z|x,y) \| p_\theta(z)] - D_{KL}[q_\varphi(z|x,y) \| q_\lambda(z|x)] - D_{KL}[q_\varphi(z|x,y) \| q_\lambda(z|y)]$$

元の目的関数　　単一モダリティのエンコーダの近似の項

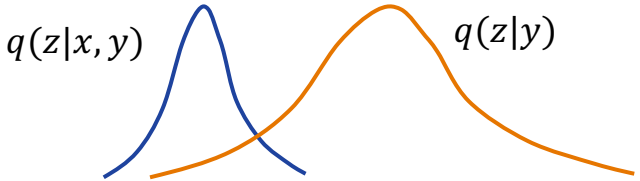


- $q(z|x)$と$q(z|y)$を単一モダリティ入力から潜在変数への写像に用いる．
  - 欠損補完をせずに潜在変数を推論できるようになる（反復サンプリングが必要ない）．

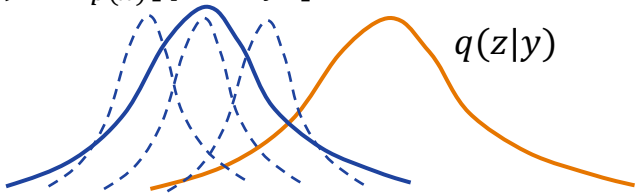
=> JMVAE [Suzuki+ 17]

JMVAEの妥当性について

- JMVAE におけるKL最小化による単一モダリティ分布の近似は，欠損補完に限らず使える手法
  - SCAN[Higgins+ 17]やGQN[Eslami+ 18]などでも同様のアプローチを採用.
- この手法の妥当性は？
  - 直感的には，近づける分布$q(z|x,y)$の方が不確かさが小さいため，尖った分布に近づいてしまう（左）.
  - ただし目的関数全体はデータ分布の期待値であり，式変形すると単一モダリティ分布のデータ分布での期待値$q_{avg}(z|y)$になる（右）.



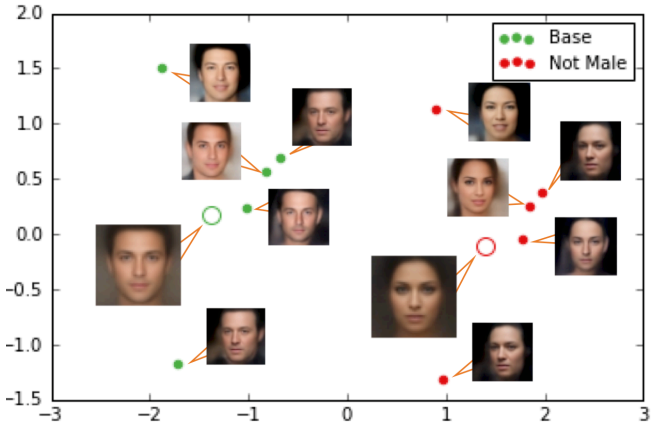
$$q_{avg}(z|y) = E_{p(x)}[q(z|x,y)]$$



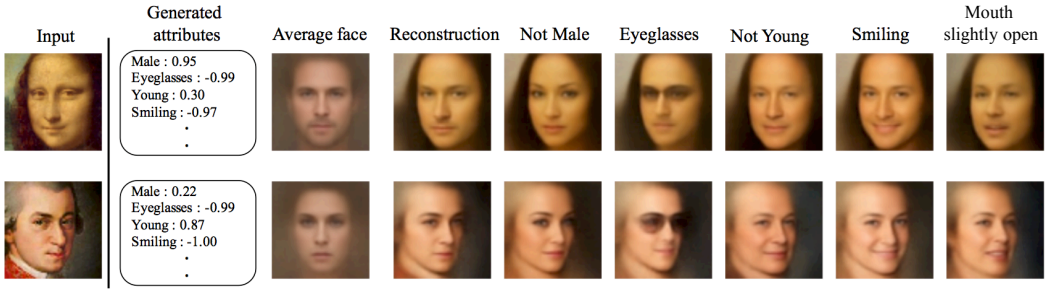
-> 明示的に単一モダリティ分布を近似する効果

JMVAE

- 双方向変換や，共有表現の学習が可能
  - e.g., 画像 (x) と属性(y)



共有表現



双方向の変換

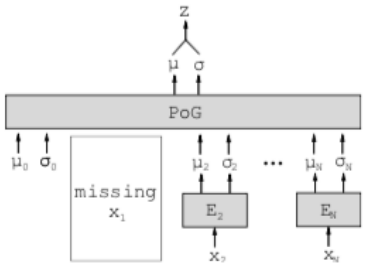
- 半教師あり学習[Suzuki+ 18]やゼロショット学習[Suzuki+ 18]などへの応用

JMVAEの発展

- JMVAEの課題
  - モダリティごとに推論モデルを用意する必要がある.
  - 3つ以上のモダリティに拡張困難.

- MVAE[Wu+ 18]
  - 異なるモダリティへの推論として，エキスパートの積（PoE）を利用
  - 追加の推論モデルが不要&任意のモダリティ数に拡張可能

$$p(z|x_1,...,x_N) \propto \frac{\prod_{i=1}^{N} p(z|x_i)}{\prod_{i=1}^{N-1} p(z)} \approx \frac{\prod_{i=1}^{N} [\tilde{q}(z|x_i)p(z)]}{\prod_{i=1}^{N-1} p(z)} = p(z)\prod_{i=1}^{N} \tilde{q}(z|x_i).$$



  - 混合エキスパート（エキスパートの和，MoE）を用いる手法もある[Kurle+ 18, Shi+ 19].

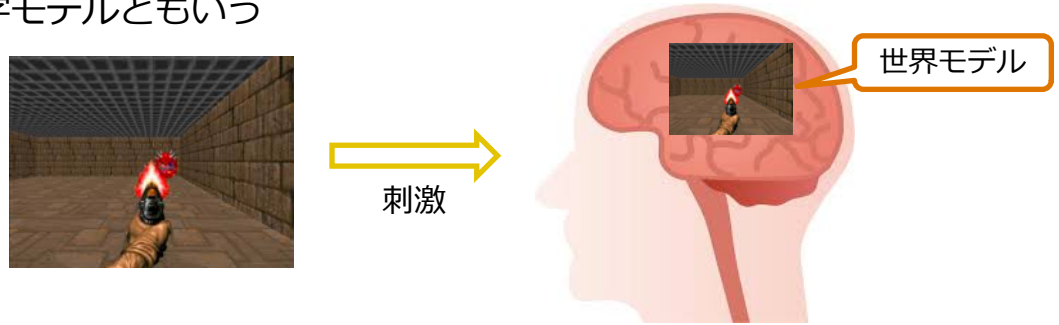
- 共有表現を学習せずに，欠損モダリティ問題を解決する手法も提案されている.
  - 各モダリティの推論モデルだけを利用[Ivanov+ 19, Jo+ 19, Tsai+ 19]

世界モデルと深層生成モデル

脳の世界モデル

- 人間は世界のあらゆるものを知覚できるわけではない.
  - 脳に入ってくる情報は非常に限られている.
  - したがって脳の内部では，限られた情報から現実世界をモデル化している.

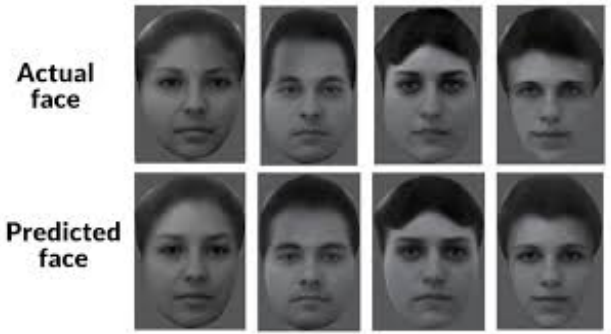
- 世界モデル（world model）：
  - 外界からの限られた刺激を元に，世界をシミュレートするモデル
  - 内部モデルや力学モデルともいう

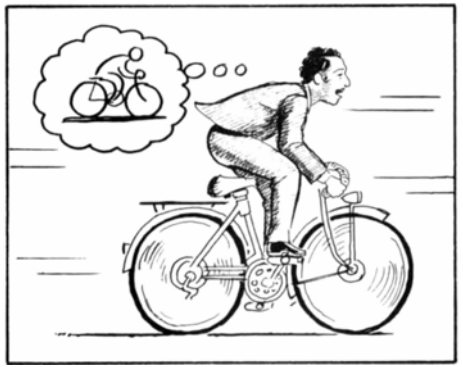


深層生成モデルによる世界モデル研究が進められている.

世界モデルにおける表現の獲得

- 世の中には大量かつ複雑な情報が溢れており，世界モデルではそれらの情報を空間的・時間的に圧縮している．
  - 脳の神経カラムの活動から顔画像を復元 [Chang+ 17]

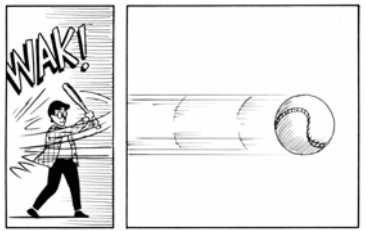




-> 世界モデルは自ら情報の構造を理解して抽象表現を獲得している

世界モデルによる予測

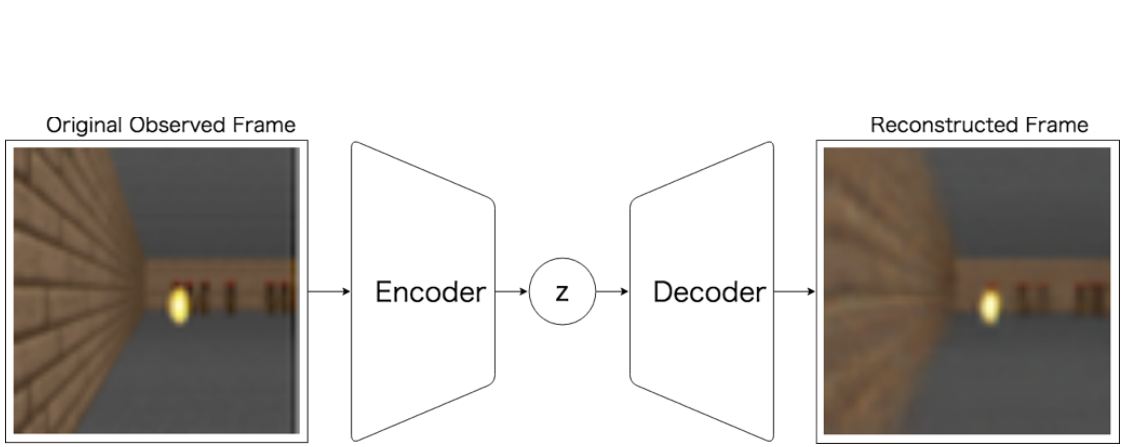
- 「今までの記憶から未来を予測する力．それが知能である．」
  - Jeff Hawkins （「On Intelligence（考える脳　考えるコンピュータ）」より）
  - より正確に言うと，現在の運動や行動を使って将来の刺激を予測している．
- 学習した脳の世界モデルによって未来をシミュレーションしている．
  - 人間はこれを常に行っていると考えられる．
- 例：バットを振ってボールに当てる
  - ボールが飛んでくる視覚情報が脳に到達する時間は，バットの振り方を決める時間よりも短い，
  - 世界モデルによって無意識に予測を行い，それにしたがって筋肉を動かしている．

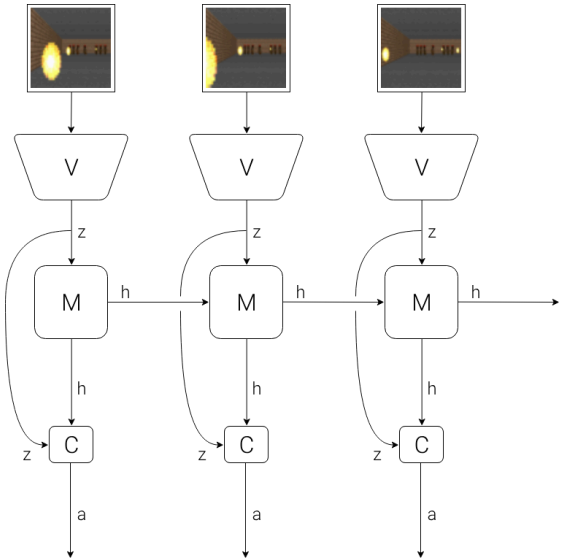


-> 知能における世界モデルの重要性

深層生成モデルを用いた世界モデル

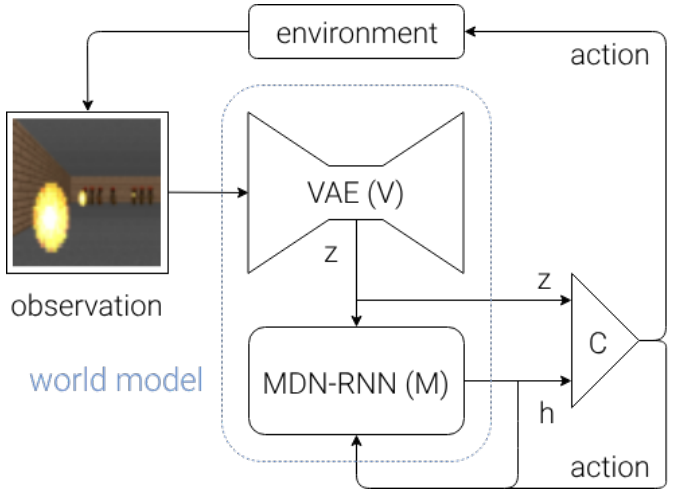
- World Model[Ha+ 18]
  - VAE+MDN-RNN[Graves + 13, Ha+ 17]で，ゲーム環境の世界モデルを学習





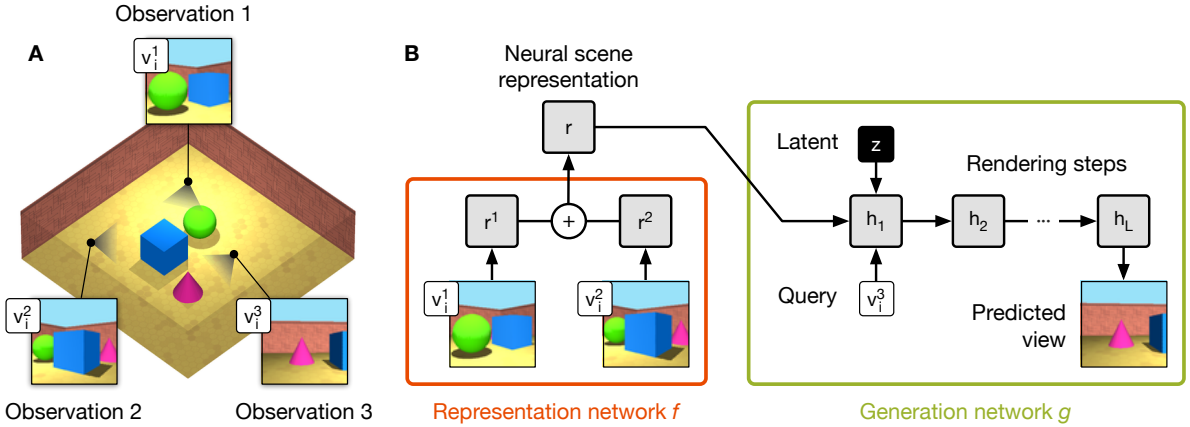
世界モデル内での強化学習

- 学習した世界モデルの中で強化学習を行う
  - 人間でいうイメージトレーニングや睡眠学習のようなもの
  - 実世界とは違い，何回でも学習できる.
- 実世界（ゲーム）でテストすると，正しく行動できていることがわかる.



より複雑な環境での世界モデル

- Generative Query Network（GQN）[Eslami+ 18]
  - ある複数の視点における画像を元に，別の視点の画像を予測する世界モデル.
  - 条件付け深層生成モデルの利用.



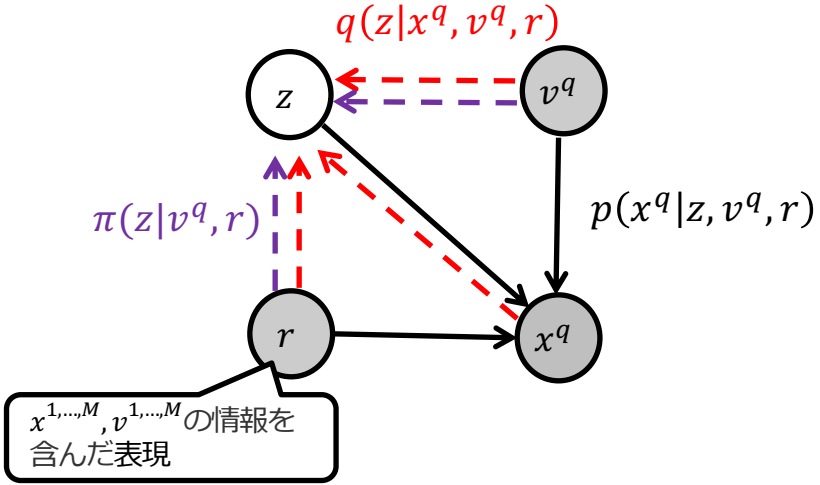
表現ネットワーク

生成ネットワーク（生成モデル）

Generative Query Network

- 表現$r = f(x^{1,\dots,M}, v^{1,\dots,M})$とクエリ視点$v^q$で条件づけた画像$x^q$の深層生成モデル（条件付けVAE）
  - rとv^qを条件付けモデルの説明でのyと考えればよい.
  - fは表現ネットワーク（M個の視点$x^{1,\dots,M}$と対応する画像$v^{1,\dots,M}$から表現rをエンコード）.

グラフィカルモデル



ELBO（目的関数）

$$\log p(x^q|v^q, r)$$

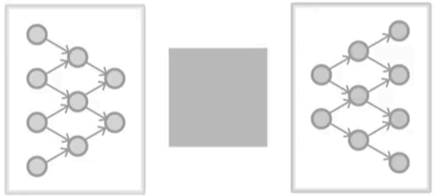
文脈とクエリ視点で条件づけた下でのクエリ画像の対数尤度

$$\geq E_{q(z|x^q, v^q, r)}\left[\log \frac{p(x^q|z, v^q, r)\pi(z|v^q, r)}{q(z|x^q, v^q, r)}\right]$$

$$= E_{q(z|x^q, v^q, r)}[\log p(x^q|z, v^q, r)] - D_{KL}[q(z|x^q, v^q, r)||\pi(z|v^q, r)]$$

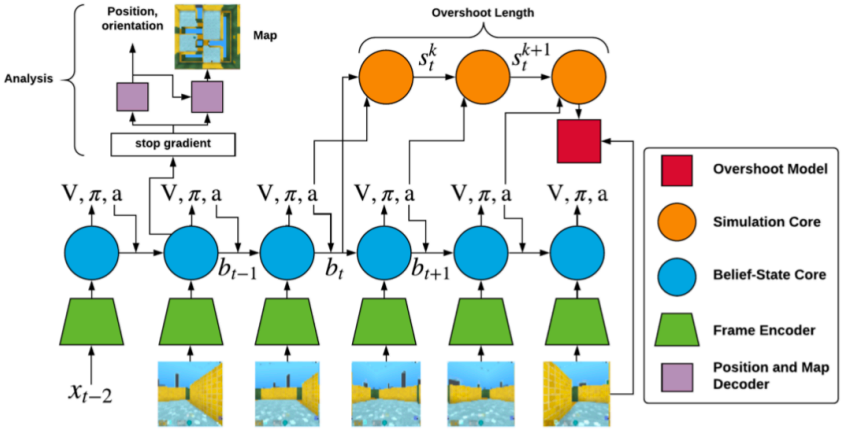
Generative Query Network

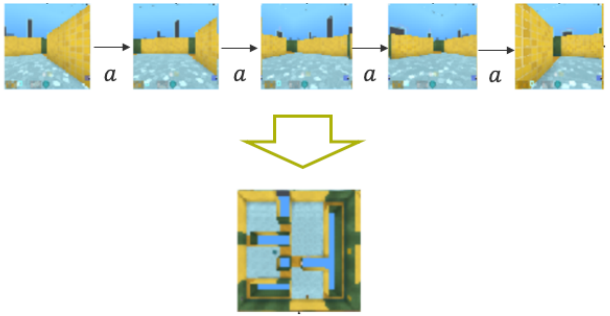
https://www.youtube.com/watch?v=RBJFngN33Qo



系列情報からの世界モデルの獲得

- Shaping Belief States with Generative Environment Models for RL [Gregor+ 19]
  - エージェントの視点と行動だけから，世界の不変的な構造を学習.
  - 信念状態を学習する予測モデル.
    - 信念の学習を無視しないようにするために，オーバーシューティングを導入
  - 信念状態からデコードすることで，その時点での地図を可視化できる.



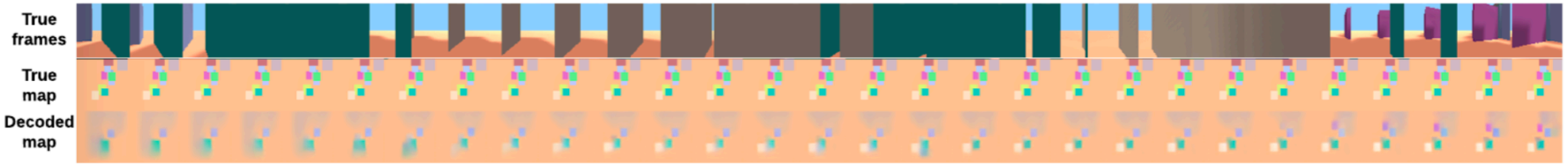


Ali Eslami
@arkitus

If you've been wondering how Generative Query Networks should be trained in the absence of camera positions (e.g. in a SLAM setting), this paper offers a possible solution. Very cool work!

信念状態の可視化と未来の予測

- 信念状態のデコード



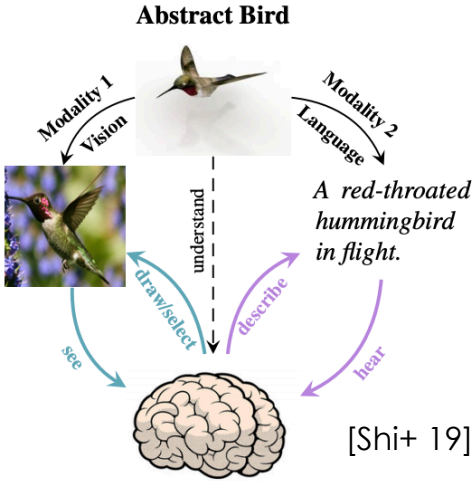
  - エージェントが初めて見たものは地図上に表示される.
  - 新しい情報が入ってきてもこれまでの情報は消えずに保たれている.

- ある程度環境内を回った後（170ステップ）にロールアウト（未来の予測）

  - 正しくシミュレートできている-> 一貫した系列予測が可能

世界モデルとマルチモーダル学習

- 従来の世界モデル研究では，単一モダリティ（主に画像）のみを扱っていた．
  - 一方で，人間は様々なモダリティ情報を元に，脳内に抽象的な世界をモデル化している．

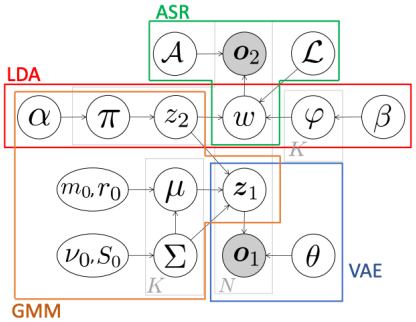


[Shi+ 19]

⇒ 世界モデルにおけるマルチモーダル学習の重要性

Neuro-SERKET

- Neuro-SERKET [Taniguchi+ 19]
  - 深層生成モデルを含む確率的生成モデルによるマルチモーダル認知アーキテクチャの提案
    - 確率モデルの結合には，SERKET[Nakamura+ 17]の方法を応用.
  - VAE+GMM+LDA+ASRの例：
    - 数字の画像（MNIST）と音声から学習



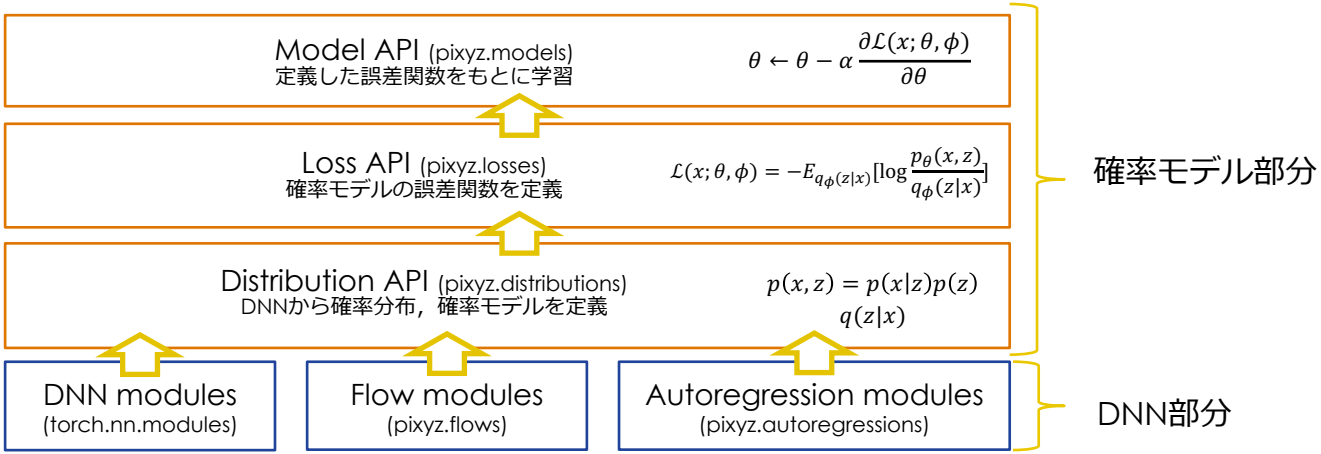
model	Accuracy (%)		Features introduced in Neuro-SERKET		
	GMM	LDA	Head-to-head	Tail-to-tail	Neural net
VAE GMM LDA ASR	63.7	27.5			
VAE GMM LDA+ASR	63.7	92.7	✓		
VAE+GMM LDA+ASR	66.7	92.7	✓		✓
VAE+GMM+LDA+ASR	**91.0**	**93.7**	✓	✓	✓

- 今後の課題：
  - 複雑な認知アーキテクチャを全て深層生成モデルで設計・学習することは可能か？
  - そうした複雑な深層生成モデルを簡潔に実装することは可能か？

Pixyz

- Pixyz [Suzuki+ 18]：深層生成モデルに特化した確率プログラミングライブラリ
  - 複雑な深層生成モデルを簡潔に実装できることが特徴
  - PyTorch（深層学習ライブラリ）ベース

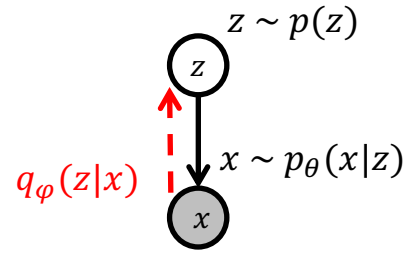
- 直感的な実装を実現するために，DNNと確率モデルの設計が分離していることが特徴.
  - DNNを意識せずに確率モデルを実装に集中できる.
  - Pixyzの構成図（実装の流れ）：



Pixyzの実装例

- Variational autoencoder (VAE) ：

$$-E_{p_{data}(x)}[D_{KL}[q_\varphi(z|x) \parallel p(z)] + E_{q_\varphi(z|x)}[\log p_\theta(x|z)]]$$



PyTorch

```
class VAE(nn.Module):

    def __init__(self):
        super(VAE, self).__init__()

        self.fc1 = nn.Linear(784, 512)
        self.fc21 = nn.Linear(512, 2)
        self.fc22 = nn.Linear(512, 2)

        self.fc3 = nn.Linear(2, 512)
        self.fc4 = nn.Linear(512, 784)

        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def encode(self, x):
        h = self.relu(self.fc1(x))
        return self.fc21(h), self.fc22(h)

    def reparameterize(self, mu, logvar):
        if self.training:
            std = logvar.mul(0.5).exp_()
            eps = Variable(std.data.new(std.size()).normal_())
            return eps.mul(std).add_(mu)
        else:
            return mu

    def decode(self, z):
        h = self.relu(self.fc3(z))
        return self.sigmoid(self.fc4(h))

    def forward(self, x):
        x = x.view(-1, 784)
        mu, logvar = self.encode(x)
        z = self.reparameterize(mu, logvar)
        return self.decode(z), mu, logvar
```

Pixyz

```
class Inference(Normal):
    def __init__(self):
        super(Inference, self).__init__(cond_var=["x"], var=["z"], name="q")
        self.model = nn.Sequential(nn.Linear(2, 512), nn.ReLU())
        self.loc, self.scale = nn.Linear(512, z_dim), nn.Linear(512, z_dim)
    def forward(self, x):
        return {"loc": self.loc(self.model(x)), "scale": F.softplus(self.scale(self.model(x)))}
q = Inference()
```

```
class Generator(Bernoulli):
    def __init__(self):
        super(Generator, self).__init__(cond_var=["z"], var=["x"], name="p")
        self.model = nn.Sequential(nn.Linear(z_dim, 512), nn.ReLU(), nn.Linear(512, x_dim))
    def forward(self, z):
        return {"probs": torch.sigmoid(self.model(x))}
p = Generator()
```

DNN部分（確率分布を定義）

```
elbo = (-KullbackLeibler(q, prior) + E(q, LogProb(p))).mean()
```

確率モデル部分
（1行で書ける）

-> 確率モデルの式と対応した簡潔な実装

複雑な深層生成モデルの実装

- TD-VAE[Gregor+ 18]：深層生成モデルによる時系列モデルの一つ

$$\mathop{\mathbb{E}}_{q\left(z_{t_1}, z_{t_2} | b_{t_1}, b_{t_2}\right)} \left[\log p\left(x_{t_2} | z_{t_2}\right) + \log p_B\left(z_{t_1} | b_{t_1}\right) + \log p\left(z_{t_2} | z_{t_1}\right) - \log p_B\left(z_{t_2} | b_{t_2}\right) - \log q\left(z_{t_1} | z_{t_2}, b_{t_1}, b_{t_2}\right)\right]$$

  - 従来の深層確率プログラミング言語では実装困難

- Pixyzでの実装（確率モデリング部分のみ表示）

```
kl = KullbackLeibler(q, p_b1)
reconst = E(q,  -p_t.log_prob() - p_d.log_prob() + p_b2.log_prob())
step_loss = E(p_b2, reconst + kl)
_loss = IterativeLoss(step_loss, max_iter=seq_len-1,
                      series_var=["x", "b"], timestep_var=["t"],
                      slice_step=slice_step)
loss_cls = E(belief_state_net, _loss).mean()
print_latex(loss_cls)
```

$$mean\left(\mathbb{E}_{p(b|x)}\left[\sum_{t=1}^{19} \mathbb{E}_{f(x_{t2}, b_{t1}, b_{t2}|t,x,b)}\left[\mathbb{E}_{p_b(z_{t2}|b_{t2})}\left[D_{KL}\left[q(z_{t1}|z_{t2}, b_{t1}, b_{t2})||p_b(z_{t1}|b_{t1})\right] + \mathbb{E}_{q(z_{t1}|z_{t2}b_{t1}, b_{t2})}\left[\log p_b(z_{t2}|b_{t2}) - \log p_d(x_{t2}|z_{t2}) - \log p_t(z_{t2}|z_{t1})\right]\right]\right]\right)$$

実装したモデルは
式として可視化できる

Pxyzの学習速度

- PyTorch上に実装されている既存の確率プログラミング言語Pyroと比較.
  - PyTorchでの実装（確率プログラミング言語を使わない実装）とも比較.
  - 1ステップあたりのVAEの学習時間の比較（z：潜在変数の次元，h：隠れ層の次元）

$\#z$	$\#h$	PyTorch (ms)	Pyro (ms)	Pixyz (ms)
10	400	2.47 ± 0.11	4.91 ± 0.12	3.61 ± 0.11
30	400	2.49 ± 0.10	4.94 ± 0.13	3.58 ± 0.10
10	2000	3.26 ± 0.11	4.93 ± 0.12	3.62 ± 0.09
30	2000	3.28 ± 0.10	4.95 ± 0.12	3.65 ± 0.09

- 結果：
  - Pyroと比べて高速.
  - PyTorchでの実装と比べても大きく速度が落ちていない.

速度面からも，Pixyzは複雑な深層生成モデルの実装に適している

Pixyzの公開・利用状況

- 公開・利用状況
  - リポジトリ：https://github.com/masa-su/pixyz
  - Pixyzを用いた様々な深層生成モデルを実装
    - Pixyzoo（ https://github.com/masa-su/pixyzoo ）
  - GQN[Eslami+ 18]の再現実装（by 松尾研 谷口くん）.
    - GQN第一著者にも紹介される
- 今後の予定
  - 時系列モデルの直感的な取り扱い
  - 実装例の充実（現在約25種類のモデルの実装例を公開中）
  - チュートリアル・ホームページの立ち上げ・整備
  - 他ライブラリとの連携

Neural Scene Representation and Rendering

S. M. Ali Eslami*, Danilo J. Rezende*, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, Demis Hassabis

Science, 2018.

Overview, Video, Science PDF, Preprint PDF, Datasets, Related work

Implementations of the Generative Query Network:

- PyTorch (by Jens Peterson)
- PyTorch (by Shohei Taniguchi)
- PyTorch (by Jesper Wohlert)
- Pixyz (by Shohei Taniguchi and Masahiro Suzuki)
- Chainer (by @musyokudon)
- Tensorflow (by Oliver Groth and Ștefan Săftescu)

-> 世界モデルの記述言語としてのPixyz

まとめ

- 深層生成モデルと世界モデルについて話しました.
  - マルチモーダル学習との関係やPixyzについても紹介.

- 参考（今回の内容をより詳しく説明したスライド）：
  - GAN（と強化学習との関係）
    - https://www.slideshare.net/masa_s/gan-83975514
  - 「世界モデル」と関連研究について
    - https://www.slideshare.net/masa_s/ss-97848402
  - GQNと関連研究，世界モデルとの関係について
    - https://www.slideshare.net/DeepLearningJP2016/dlgan-111725780